SOFTWARE PROGRAM

PRODUCTS

6 5 0 2 F O R T H V E R 1 . 2 S Y S T E M

PRELIMINARY REFERENCE MANUAL

FEBRUARY 28, 1979

DEVELOPED BY:

PROGRAMMA INTERNATIONAL, INC.

3400 WILSHIRE BOULEVARD

LOS ANGELES, CA 90010

1. INTRODUCTION.

6502FORTH IS A UNIQUE THREADED LANGUAGE THAT IS IDEALLY SUITED FOR MICROPROCESSOR SYSTEMS. PROGRAMS WRITTEN IN 6502FORTH ARE COMPACT; I.E. IN 6K TO 8K BYTES, THE USER MAY HAVE THE INTERACTIVE 6502FORTH COMPILER/INTERPRETER RUNNING STAND-ALONE USING THE SYSTEM'S MONITOR FOR I/O, AND OTHER RUN-TIME ROUTINES, PLUS AN INCREMENTAL ASSEMBLER, CASSETTE MEMORY SOFTWARE, AND A TEXT EDITOR. NOT ONLY DOES ALL OF THIS FIT INTO THE 4K TO 6K BYTES (80% OF WHICH IS WRITTEN IN 65D2FORTH), BUT ALSO, IT RUNS IN THE SAME SPACE WITH NO ADDITIONAL SYMBOL TABLE AREA, OVERLAYS, SWAPPING, OR USE OF ANY OTHER SOFTWARE.

WHILE 6502FORTH GIVES ALL OF THE CONVENIENCES OF INTERACTIVE INTERPRETERS, IT IS ALSO VERY FAST. FOR MOST APPLICATIONS, THE RUN-TIME OVERHEAD IS 70 TO 100 PERCENT FOR MICROCOMPUTERS, AS COMPARED TO 1000 PERCENT OR MORE WHICH IS COMMON FOR INTERPRETERS SUCH AS BASIC. NUMBER CRUNCHING APPLICATIONS IN 65D2FORTH MAY TAKE MUCH LONGER; HOWEVER, IF 6502FORTH IS NOT FAST ENOUGH, THE USER MAY CHOOSE TO USE THE SYSTEM'S OWN ASSEMBLER TO RE-CODE INNER LOOPS.

ONE OF THE BEST ADVANTAGES OF 6502FORTH OVER OTHER PROGRAMMING LANGUAGES IS THAT SOFTWARE DEVELOPMENT TIMES ARE CUT IN HALF OR MUCH MORE OVER ASSEMBLY LANGUAGE PROGRAMMING. THE PROGRAMMING IN 6502FORTH IS ENTIRELY DONE IN A STRUCTURED MANNER (THERE IS NO GOTO), AND THE RESULTING CODE IS RE-ENTRANT AND CAN BE DESIGNED FOR PROM.

THE 6502FORTH IMPLEMENTATION OF THE FORTH LANGUAGE REQUIRES A MACHINE CONFIGURATION THAT CONTAINS BOTH AN INPUT KEYBOARD DEVICE AND AN OUTPUT DISPLAY DEVICE. A RECOMMENDED MEMORY CONFIGURATION IS 16K BYTES., A DISK DEVICE IS NICE FOR STORING 6502FORTH SOURCE PROGRAMS, BUT THE AUDIO CASSETTE RECORDER (OR SIMILAR SEQUENTIAL UNIT) SUFFICES TO SIMULATE THE VIRTUAL MEMORY STORAGE OF SOURCE PROGRAMS. 65D2FORTH WORKS WELL WITH A CRT VIDEO DISPLAY UNIT, SO HARD COPY IS NOT NECESSARY.

THE FORTH LANGUAGE HAS EXISTED FOR SEVERAL YEARS, AND IS USED COMMERCIALLY IN A NUMBER OF INSTALLATIONS. UNTIL RECENTLY, IT HAS BEEN PRICED FAR OUT OF THE REACH OF THE AMATEUR HOBBYIST. MOST COMPUTER PROFESSIONALS HAVE NEVER HEARD OF IT. THE FORTH LANGUAGE, HOWEVER, IS IN THE PUBLIC DOMAIN, AND 6502FORTH IS THE FIRST IMPLEMENTATION ON THE 6502 MICROPROCESSOR UNIT.

THE ORIGINAL FORTH LANGUAGE WAS FIRST DEVELOPED BY CHARLES H. MOORE AT THE NATIONAL RADIO ASTRONOMY OBSERVATORY UNDER CONTRACT WITH THE NATIONAL SCIENCE FOUNDATION. A PAPER IN THE JOURNAL OF ASTRONOMY AND ASTROPHYSICS SUPPLEMENT (1974, 15, 497-511) TITLED "FORTH: A NEW WAY TO PROGRAM A MINI-COMPUTER", BY CHARLES H. MOORE, DESCRIBES THE ORIGINAL SPECIFICATIONS AND DESCRIPTION OF THE FORTH LANGUAGE.

2. OVERVIEW.

THE BASIC ELEMENT OF THE 6502FORTH SYSTEM IS TERMED A "WORD", WHICH IS ROUGHLY COMPARABLE TO A SUBROUTINE. A 6502FORTH WORD, WHEN REFERENCED (OR EXECUTED), CAUSES AN ACTION OR SEQUENCE OF ACTIONS TO BE PERFORMED. THEREFORE, WHEN A WORD IS EXECUTED, A SUBROUTINE IS CALLED AND THE VARIOUS ACTIONS INDICATED BY THE SUBROUTINE OCCUR. BEFORE A WORD CAN BE EXECUTED, IT MUST HAVE BEEN PREVIOUSLY DEFINED AND STORED IN THE 6502FORTH "DICTIONARY". THE DICTIONARY IS A LINKED LIST OF WORDS TOGETHER WITH THEIR MEANINGS OR ACTIONS. THE ACTIONS MAY BE EXPRESSED AS MACHINE-LANGUAGE INSTRUCTIONS OR AS A SEQUENCE OF OTHER WORDS. THE 6502FORTH DICTIONARY INITIALLY CONTAINS AROUND 200 WORDS, WHICH ARE REFERRED TO AS THE "STANDARD VOCABULARY". SOME OF THESE WORDS CAN BE USED TO DEFINE NEW WORDS. WRITING A 6502FORTH SOURCE PROGRAM CONSISTS OF DEFINING A SERIES OF NEW WORDS IN TERMS OF THE OLD DEFINITIONS.

A 6502FORTH USER AT THE KEYBOARD TERMINAL MAY TYPE WORDS INTO THE COMPUTER. ANY SEQUENCE OF CHARACTERS MAY BE USED TO DEFINE A WORD. THE ONLY RESERVED CHARACTERS ARE THOSE THAT HAVE SPECIAL MEANING FOR THE MACHINE ENVIRONMENT THAT'S BEING USED. OTHERWISE, ANY COMBINATION OF LETTERS, NUMBERS, AND SPECIAL CHARACTERS CAN BE USED IN DEFINING THE NAME OF A 6502FORTH WORD. A WORD MUST BE SEPARATED FROM OTHER WORDS BY A DELIMITER CHARACTER. THE DELIMITER CHARACTER IS NORMALLY A SPACE OR BLANK. INPUT FROM THE KEYBOARD TERMINAL IS "BUFFERED" BY THE 6502FORTH SYSTEM, AND CONTROL PASSES TO THE SYSTEM FOR EXECUTION WHEN THE "CARRIAGE RETURN" KEY IS DEPRESSED. FOR EXAMPLE, THE INPUT STREAM:

```
7 3 + . CR
```

WILL CAUSE THE NUMBERS 7 AND 3 TO BE ADDED TOGETHER AND A RESULT OF A (ASSUMING BASE 16) TO BE PRINTED ON THE OUTPUT DEVICE. THE 6502FORTH SYSTEM WILL THEN DO A "CARRIAGE RETURN" AND "LINE FEED" AND PROCEED TO PROMPT THE USER FOR FURTHER INPUT.

IN ORDER TO CONSERVE COMPUTER MEMORY, NOT ALL OF THE CHARACTERS IN A NAME ARE STORED. IN THE 6502FORTH IMPLEMENTATION, A NAME IS RECOGNIZED ON THE BASIS OF THE FIRST FOUR (4) CHARACTERS.

REVERSE POLISH NOTATION (RPN) AND LAST-IN FIRST-OUT STACKS (LIFO), SUCH AS THOSE USED IN HEWLETT-PACKARD CALCULATORS, ARE USED IN THE 6502FORTH SYSTEM. THEREFORE, TO FURTHER EXPLAIN THE PREVIOUS EXAMPLE IN DETAIL: THE NUMBER 7 WAS PUSHED ONTO THE STACK, AND IT WAS FOLLOWED BY THE NUMBER 3. BOTH NUMBERS WERE THEN ADDED TOGETHER AND "POPPED" OFF THE STACK BY THE PREVIOUSLY DEFINED WORD "+". THE RESULT OF 10 IS "PUSHED" ONTO THE STACK BY THE "+" OPERATION ALSO. THE WORD "." THEN "POPS" THE STACK TO ITS INITIAL CONDITION AND PRINTS THE NUMBER 10 ON THE OUTPUT LIST DEVICE.

IF A WORD THAT IS TYPED IN THE INPUT STREAM CANNOT BE LOCATED IN THE 6502FORTH DICTIONARY, THE SYSTEM ATTEMPTS TO TREAT IT AS A NUMBER. IF THIS IS POSSIBLE, THAT IS, IF THE WORD IS ACTUALLY A NUMBER IN THE PROPER FORMAT AND BASE, THEN THE NUMBER IS CONVERTED TO BINARY AND MADE AVAILABLE FOR FURTHER PROCESSING. IF THE WORD CANNOT BE INTERPRETED AS A NUMBER, OR IF CONVERSION IS NOT POSSIBLE, THEN 6502FORTH WILL ISSUE ITS STANDARD ERROR "+++ERROR XX", WHERE XX IS THE ERROR CODE.

WORDS CAN BE ADDED TO THE 6502FORTH DICTIONARY IN SEVERAL WAYS. AS WITH ANY PROGRAMMING LANGUAGE OR NOTATION, THE FASTEST ROUTE TO FLUENCY IS THROUGH EXAMPLES AND HANDS-ON USAGE. THEREFORE, IF THE INPUT STREAM CONSISTS OF:

```
2 VARI VALU
```

6502FORTH DEFINES A NEW WORD IN THE DICTIONARY CALLED "VALU", WHICH IS A 16-BIT (TWO BYTE) VARIABLE, WHOSE VALUE IS PRESET TO "2". REMEMBER THAT 6502FORTH ONLY LOOKS AT (AND ONLY REMEMBERS) THE FIRST 4 CHARACTERS OF A WORD. THE INPUT STREAM:

```
2 VARIABLE VALUE
```

WOULD PRODUCE EXACTLY THE SAME RESULTS. CONTINUING WITH THE SAME EXAMPLE, THE INPUT STREAM:

```
VALU @ .
```

CAUSES THE ADDRESS OF "VALU" TO BE PUSHED ONTO THE STACK. THE "@" THEN REPLACES THE ADDRESS ON THE STACK WITH THE CONTENTS FOUND AT THAT ADDRESS. THE "." CAUSES THE ENTRY ON THE STACK TO BE PRINTED ON THE OUTPUT DEVICE, HENCE, A TWO (2) WOULD BE PRINTED ON THE OUTPUT DEVICE.

WORDS THAT ARE ALREADY IN THE 6502FORTH DICTIONARY MAY BE USED TO FORM OTHER NEW WORDS USING THE ":" WORD. FOR EXAMPLE:

```
: ? @ . ;
```

THIS INPUT STREAM DEFINES A NEW WORD CALLED "?", WHICH WHEN EXECUTED CAUSES THE WORD "@" AND THE WORD "." TO BE EXECUTED. THE ";" WORD IS THE 6502FORTH WORD THAT TERMINATES THE DEFINITION MODE. WITH THE ABOVE NEW WORD, THE USER CAN NOW INPUT:

```
VALU ?
```

WHICH WILL CAUSE THE VALUE OF THE VARIABLE "VALU" TO BE PRINTED ON THE OUTPUT DEVICE. IN THIS CASE A TWO (2) WOULD BE PRINTED.

THE WORDS THAT COMPOSE THE "STANDARD VOCABULARY" ARE LISTED IN THE SECTION TITLED "STANDARD VOCABULARY". THE ACTIONS OF THE STANDARD VOCABULARY WORDS ARE ALSO EXPLAINED IN THAT SECTION.

INPUT TO THE 6502FORTH SYSTEM CAN ALSO COME FROM A BLOCK BUFFER INSTEAD OF THE KEYBOARD. THIS BUFFER NORMALLY CONTAINS ASCII CHARACTERS THAT HAVE BEEN PREVIOUSLY STORED ON A MASS STORAGE DEVICE.

3. STACKS.

NUMBERS AND OTHER DATA ARE NORMALLY HANDLED THROUGH THE 6502FORTH "NORMAL STACK" (PARAMETER STACK). THIS IS A "PUSH-DOWN" STACK THAT USES THE "LAST-IN FIRST-OUT" (LIFO) TECHNIQUE. A PUSH-DOWN STACK IS A STORAGE MANAGEMENT STRUCTURE IN WHICH A NEW DATA ITEM MAY BE STORED (PUSHED) ON TOP OF OLDER DATA ITEMS. THE ITEM ON TOP OF THE STACK MAY BE RETRIEVED BY "POPPING" THE STACK.

ONE ADVANTAGE OF A "PUSH-DOWN" STACK IS THAT FIXED STORAGE LOCATIONS IN MEMORY NEED NOT BE ASSIGNED FOR TEMPORARY DATA. HENCE, STORAGE IS CONSERVED AND THE PROGRAMMER'S "BOOKKEEPING" EFFORT IS REDUCED.

MOST 6502FORTH WORDS, WHICH OPERATE ON DATA, ACCEPT THEIR DATA FROM THE NORMAL STACK, OPERATE ON THEM, AND THEN PUSH THE RESULTS BACK ONTO THE STACK. THEREFORE, ARITHMETIC EXPRESSIONS MUST BE SPECIFIED IN "REVERSE POLISH NOTATION" (I.E. WITH OPERANDS PRECEEDING OPERATORS).

6502FORTH USERS HAVE THE OPTION OF SPECIFYING THE LOCATIONS WHERE THE DICTIONARY, THE NORMAL STACK, THE RETURN STACK, AND THE BUFFERS ARE TO BE LOCATED. A LOGICAL END OF MEMORY CAN BE SET SO THAT THE DICTIONARY NEVER EXCEEDS A SPECIFIED LOCATION. IT IS RECOMMENDED THAT THE USER USE THE SUGGESTED LOCATIONS FOR EACH OF THE ABOVE AREAS. THE SUGGESTED VALUES ARE DESCRIBED IN THE APPENDIX THAT CORRESPONDS TO THE MACHINE CONFIGURATION BEING USED.

THERE ARE THREE (3) STACKS USED BY THE 6502FORTH SYSTEM. ALL OF THEM USE THE LAST-IN FIRST-OUT (LIFO) TECHNIQUE. ASSUMING THE SUGGESTED VALUES ARE USED FOR MEMORY ALLOCATION, THE "NORMAL STACK" IS VARIABLE IN LENGTH (DEPENDING ON THE RAM SIZE AVAILABLE); AND IT GROWS DOWNWARD TOWARD THE FORTH DICTIONARY, WHICH IN TURN GROWS UPWARD. THE FORTH WORD "SP" (STACK POINTER) IS A CONSTANT WHICH CONTAINS THE ADDRESS OF THE TOP (CURRENT) STACK VALUE. THIS ADDRESS POINTS TO THE LEAST SIGNIFICANT BYTE (LSB) OF THE 16-BIT STACK VALUE.

THE "RETURN STACK" IS VARIABLE IN LENGTH AND ITS MAXIMUM BOUNDARY ORIGINATES ON THE SAME LOCATION AS THE NORMAL STACK. THE RETURN STACK GROWS TOWARD LOW MEMORY IN THE SAME MANNER AS THE NORMAL STACK. THE FORTH WORD "RS" (RETURN STACK) IS A CONSTANT WHICH CONTAINS THE ADDRESS OF THE TOP (CURRENT) RETURN STACK VALUE. THIS ADDRESS ALSO POINTS TO THE LSB OF THE 16-BIT VALUE. THE RETURN STACK IS USED PRIMARILY BY THE FORTH SYSTEM FOR LOOP PROCESSING.

THE THIRD STACK IS THE "HARDWARE STACK" USED BY THE 6502 MICROPROCESSOR. IT IS LOCATED AT LOCATION $01FF, AND IT GROWS DOWNWARD TOWARD LOW MEMORY. ITS LOCATION CANNOT BE CHANGED BY THE 6502FORTH USER. THIS STACK IS NOT NORMALLY USED BY 6502FORTH PROGRAMMERS, SO THERE IS NO FORTH WORD THAT CONTAINS ITS ADDRESS.

4. DICTIONARY.

THE 6502FORTH DICTIONARY IS A LINKED LIST OF WORDS. THE DICTIONARY NORMALLY BEGINS FROM THE END OF THE 6502FORTH NUCLEUS AND GROWS TOWARD HIGH MEMORY. THE DICTIONARY CONTAINS ALL OF THE 6502FORTH WORDS AVAILABLE TO THE USER.

THERE ARE THREE (3) GROUPS OF INFORMATION THAT CAN BE FOUND WITHIN EACH WORD IN THE DICTIONARY. THE FIRST FOUR BYTES OF ANY DICTIONARY WORD CONTAIN THE NAME OF THE WORD IN ASCII CODE. THE MOST SIGNIFICANT BIT OF THE FIRST BYTE OF THE WORD MAY BE SET TO ONE (1) TO INDICATE TO THE 6502FORTH SYSTEM THAT THIS IS AN IMMEDIATE WORD. OF ALL THE DICTIONARY WORDS, IMMEDIATES ARE THOSE THAT ARE EXECUTED DIRECTLY WHEN FOUND IN THE INPUT STREAM. NAMES ARE CONSIDERED EQUIVALENT IF THEIR FIRST FOUR CHARACTERS ARE THE SAME. NAMES THAT HAVE LESS THAN FOUR CHARACTERS ARE AUTOMATICALLY PADDED WITH SPACES OR BLANKS.

THE NEXT TWO BYTES OF A WORD FOLLOWING THE NAME CONTAIN THE ADDRESS OF THE FIRST BYTE OF THE PREVIOUS DICTIONARY ENTRY. THESE TWO BYTES ARE USED TO LINK THE DICTIONARY. THE LINK ADDRESS OF THE FIRST WORD IN THE DICTIONARY IS SET TO ZERO (0000), AND INDICATES THE BEGINNING OF THE DICTIONARY. THIS IS ALSO THE END OF THE CHAIN OF LINKED DICTIONARY WORDS, BECAUSE THE DICTIONARY IS SEARCHED BACKWARDS (FROM THE LAST WORD ENTERED TO THE FIRST). THESE FIRST SIX BYTES (THE NAME AND THE LINK POINTER) ARE COMMONLY REFERRED TO AS THE "HEADER".

THE REMAINING BYTES OF AN ENTRY IN THE DICTIONARY CONSISTS OF MACHINE LANGUAGE CODE, WHICH IS REALLY A SUBROUTINE. HENCE, THE WORD IS EXECUTED (BY THE FORTH NUCLEUS) BY DOING A JUMP TO SUBROUTINE (JSR) TO THE FIRST BYTE OF MACHINE CODE. THE MACHINE LANGUAGE CODE NORMALLY TERMINATES WITH A RETURN FROM SUBROUTINE (RTS) INSTRUCTION.

FOR EXAMPLE:

```
: ABC PAGE CR ;
```

WILL GENERATE THE FOLLOWING DICTIONARY ENTRY:

ADDRESS	CONTENTS	COMMENTS
1000	$41	"A" HEADER START
1001	42	"B"
1002	43	"C"
1003	20	" " PADDING OF A BLANK CHARACTER
1004	0E	LSB OF LINK TO NEXT WORD HEADER
1005	08	MSB OF LINK TO NEXT WORD HEADER
1006	20	JSR INSTRUCTION OP CODE
1007	00	LSB OF ADDRESS OF CLR WORD SUBROUTINE
1008	0A	MSB OF ADDRESS OF CLR WORD SUBROUTINE
1009	20	JSR INSTRUCTION OP CODE
100A	FF	LSB OF ADDRESS OF CR WORD SUBROUTINE
100B	08	MSB OF ADDRESS OF CR WORD SUBROUTINE
100C	60	RTS INSTRUCTION OP CODE

5. BLOCK STORAGE.

MOST PRACTICAL APPLICATIONS OF THE 6502FORTH LANGUAGE REQUIRE AN AUXILIARY STORAGE DEVICE. "FLOPPY DISKS" ARE USUALLY PREFERABLE, HOWEVER, SEQUENTIAL MAGNETIC TAPE CASSETTES SUFFICE FOR THE AVERAGE USER.

BLOCK STORAGE IS USED AS A "VIRTUAL MEMORY" SCHEME, WHERE ONE MAY STORE DATA IN BLOCKS WHEN THERE IS INSUFFICIENT SPACE IN THE COMPUTER'S MAIN MEMORY. BLOCKS ARE SUITABLE TO STORE LARGE AMOUNTS OF DATA. NORMALLY, IN THE 6502FORTH SYSTEM, THIS DATA IS THE SOURCE TEXT.

BLOCKS ARE USUALLY CALLED "SCREENS", AND ARE NUMBERED SEQUENTIALLY BEGINNING AT ONE (1). TWO LARGE BUFFERS (BUFFER 0 AND BUFFER 1) ARE SET ASIDE TO ACT AS THE HOLDING AREAS WHERE ALL TEXT EDITING WILL OCCUR.

6. KERNEL.

THE KERNEL (NUCLEUS) OF THE 6502FORTH SYSTEM IS AN ASSEMBLY LANGUAGE PROGRAM WHOSE BASIC FUNCTION IS TO PROVIDE THE CAPABILITY OF STARTING AND ADDING NEW WORD ENTRIES TO THE DICTIONARY. SOME OF THE TASKS THAT ARE PERFORMED BY THE KERNEL OF THE 6502FORTH SYSTEM ARE:

1. INITIALIZING THE SYSTEM
2. BUFFERING THE INPUT FROM THE KEYBOARD
3. SEARCHING THE DICTICNARY
4. CONVERTING ASCII INPUT TO NUMBERS
5. PARSING THE INPUT BUFFER
6. EXECUTING WORDS IN THE DICTIONARY
7. ADDING WORDS TO THE DICTIONARY
8. CHECKING FOR ERRORS

THE KERNEL IS LOCATED AT A FIXED LOCATION IN MEMORY AND IS USUALLY FOLLOWED BY THE DICTIONARY. THE DICTIONARY GROWS TOWARD HIGH MEMORY. THE KERNEL USES TEMPORARY STORAGE IN PAGE ZERO (0) OF MAIN MEMORY.

7. DEFINING NEW WORDS.

THE DISTRIBUTED 6502FORTH SYSTEM COMES WITH ABOUT 200 WORDS DEFINED IN THE DICTIONARY. THESE WORDS PROVIDE THE FUNCTIONS THAT ARE COMMONLY NEEDED BY MOST APPLICATION PROGRAMS. PROGRAMMING IN THE 6502FORTH LANGUAGE ACTUALLY CONSISTS OF DEFINING NEW WORDS, WHICH DRAW UPON THE EXISTING VOCABULARY, AND WHICH IN TURN MAY BE USED TO DEFINE EVEN MORE COMPLEX APPLICATIONS.

6502FORTH PROVIDES A NUMBER OF WAYS TO DEFINE NEW WORDS INTO THE DICTIONARY. THE LANGUAGE EVEN PROVIDES A FACILITY FOR DEFINING WORDS WHOSE FUNCTION IS TO DEFINE WORDS. THERE ARE FOUR COMMON WAYS THAT MAY BE USED TO DEFINE WORDS USING THE STANDARD SYSTEM.

THE WORD ":" (COLON) IS USED TO DEFINE OTHER 6502FORTH WORDS IN TERMS OF EXISTING WORDS IN THE DICTIONARY. COLON DEFINITIONS ARE USUALLY MACHINE INDEPENDENT, SINCE EACH REFERS TO CODE DEFINITIONS OR OTHER COLON DEFINITIONS. AN EXAMPLE OF A COLON DEFINITION IS:

```
: NEW PAGE 1234 . CR ;
```

HERE THE WORD "NEW" IS DEFINED AS A SEQUENCE "PAGE", 1234, ".", AND "CR". THE WORDS ARE ASSUMED TO BE PRESENT IN THE DICTIONARY AT THE TIME THE DEFINITION TAKES PLACE. THE NUMBER (1234) WILL CAUSE CODE TO BE GENERATED THAT WILL PLACE IT ONTO THE STACK WHEN THE WORD "NEW" IS EXECUTED. SEMICOLON ";" IS A WORD THAT INDICATES THE END OF THE DEFINITION.

THE WORD " ^ " (UP_ARROW) ALLOWS THE USER TO DEFINE WORDS WHOSE ACTIONS ARE EXPRESSED DIRECTLY IN MACHINE (ASSEMBLY) LANGUAGE. THE WORDS THAT ARE USED AFTER THE UP_ARROW ARE DICTIONARY WORDS THAT CAUSE THE BINARY MACHINE CODE TO BE ADDED ONTO THE DICTIONARY. THE NAMES OF THESE WORDS HAVE BEEN CONVENIENTLY CHOSEN TO CLOSELY RESEMBLE THE MACHINE'S ASSEMBLER MNEMONICS. FOR THIS REASON, THE WORDS THAT ARE USED WITHIN THE UP_ARROW ARE MACHINE DEPENDENT, BUT THEY GIVE THE PROGRAMMER THE MEANS TO ACHIEVE MAXIMUM POSSIBLE SPEED OF EXECUTION. THE UP_ARROW IS ONE OF SEVERAL WORDS THAT CAN ONLY BE USED WITHIN A ":" AND ";" DEFINITION. FOR EXAMPLE:

```
: INY, ^ 88 LDA, # BA STA, a# ;
```

HERE THE WORD "INY," IS BEING DEFINED AS A SEQUENCE OF MACHINE INSTRUCTIONS AFTER THE UP-ARROW WORD IS DETECTED. IF "INY," WERE USED, IT WOULD GENERATE CODE IN THE DICTIONARY TO INCREMENT THE Y REGISTER.

CONSTANTS MAY BE DEFINED THROUGH THE WORD "CONS". FOR EXAMPLE:

```
1234 CONS X1
```

DEFINES THE 6502FORTH WORD X1. WHENEVER X1 IS EXECUTED, IT WILL PUSH THE CONSTANT 1234 ONTO THE STACK. THE USE OF X1 IN AN INPUT STREAM WOULD CREATE FEWER MACHINE INSTRUCTIONS THAN THE USE OF THE NUMBER 1234. HOWEVER, BOTH METHODS PRODUCE THE SAME RESULTS.

DATA MAY BE STORED IN NAMED LOCATIONS AS WELL AS ON THE STACK. THE NAMED LOCATIONS ARE IN THE DICTIONARY. THIS IS DONE BY USING THE 6502FORTH WORD "VARI". AS AN EXAMPLE:

```
1234 VARI X2
```

DEFINES THE WORD X2 WHICH IS THE NAME OF AN ADDRESS THAT CONTAINS THE INITIAL VALUE OF 1234. WHEN X2 IS EXECUTED, THE ADDRESS OF THE VALUE 1234 WILL BE PLACED ONTO THE STACK. OTHER 6502FORTH WORDS ARE AVAILABLE THAT CAN EITHER FETCH THE VALUE FROM THE ADDRESS ON THE STACK OR CHANGE THE VALUE AT THAT ADDRESS.

THE DIFFERENCE BETWEEN "CONS" AND "VARI" IS THAT "CONS" DEFINES A WORD WHICH REPRESENTS A VALUE. "VARI" DEFINES A WORD THAT REPRESENTS AN ADDRESS.

8. INPUT / OUTPUT.

UNDER NORMAL OPERATION, 6502FORTH ACQUIRES ITS INPUT FOR EXECUTION FROM THE KEYBOARD. THE SYSTEM IS USUALLY IDLE AND WAITING FOR THE USER TO TYPE A COMPLETE LINE OF WORDS. WHEN THIS IS DONE, THE SYSTEM INTERPRETS THE LINE, TRIES TO EXECUTE THE VALID WORDS, AND THEN PROCEEDS TO PROMPT THE USER FOR MORE INPUT.

6502FORTH MAY ALSO TAKE ITS INPUT FROM THE BLOCK BUFFER. THE WORD "LOAD" CAUSES THE SYSTEM TO READ A SCREEN FROM THE CASSETTE DEVICE AND LOAD IT INTO THE BLOCK BUFFER. THE INPUT THAT IS READ IS CALLED A SCREEN. IF THE USER HAS THE DISK VERSION OF 6502FORTH, HE MAY LOAD SCREENS OF TEXT FROM THE DISK UNIT BY ISSUING THE PROPER 6502FORTH DISK COMMANDS.

THE DATA IN THE BLOCK BUFFER CAN BE EDITED AND THEN EXECUTED JUST AS IF IT HAD ALL BEEN KEYED IN AT THE KEYBOARD AGAIN. THE WORD "EXEC" CAUSES THE INPUT TO COME FROM THE BLOCK BUFFER INSTEAD OF THE KEYBOARD BUFFER. THE INPUT ALWAYS STARTS FROM THE BEGINNING OF THE BLOCK BUFFER AND CONTINUES UNTIL THE FIRST OCCURANCE OF THE WORD ";S". THE ";S" WORD STOPS THE SCAN OF THE BLOCK BUFFER. ANY DATA FOUND IN THE BUFFER AFTER THE ";S" IS IGNORED. THE ";S" ALSO SWITCHES THE SYSTEM BACK INTO THE KEYBOARD MODE OF INPUT.

THE DATA IN THE BLOCK BUFFER CAN BE WRITTEN TO THE CASSETTE DEVICE WITH THE WORD "SAVE". THE USER SHOULD FIRST READY THE MASS STORAGE DEVICE. THE WORD "SAVE" IS THEN TYPED ON THE INPUT KEYBOARD, FOLLOWED BY A CARRIAGE RETURN (CR). THE "SAVE" WORD WILL AUTOMATICALLY PLACE A "S;" WORD AS THE LAST PHYSICAL WORD OF THE BUFFER. THE BUFFER IS THEN WRITTEN TO THE CASSETTE DEVICE. THE ";S" IS PLACED AT THE END OF THE DATA IN THE BUFFER AS A CONVENIENCE FOR THE SUBSEQUENT "LOAD" AND "EXEC" OF THE SCREEN. IF THE USER HAS THE DISK VERSION OF 6502FORTH, HE THEN MAY ISSUE THE PROPER 6502FORTH DISK COMMANDS TO SAVE SCREENS TO THE DISK.

9. CONDITIONAL BRANCHES.

6502FORTH PROVIDES SEVERAL TECHNIQUES FOR CONTROLLING THE FLOW OF PROGRAM EXECUTION. THE METHODS DESCRIBED IN THE FOLLOWING PARAGRAPHS AND THE EXAMPLES MUST BE USED WITHIN ":" (COLON) DEFINITIONS. ALL OF THESE DEFINITIONS ARE IMMEDIATE WORDS. (I.E. THEY ARE EXECUTED IMMEDIATELY - DURING THE COMPILATION - AND CAUSE MACHINE LANGUAGE CODE TO BE ADDED INTO THE DICTIONARY. IT IS UNDESIRABLE TO ADD ANYTHING TO THE DICTIONARY UNLESS A WORD IS BEING DEFINED).

THE SIMPLEST CONDITIONAL BRANCH IS SPECIFIED THROUGH THE USE OF THE WORDS "BEGIN" AND "END". THE "BEGIN" - "END" CONSTRUCT IS USEFUL FOR PROGRAM LOOPS, WHEN THE LOOP TERMINATION CONDITION CAN BE EXPRESSED BY LEAVING A ZERO OR NON-ZERO VALUE ON THE STACK. THE "END" WORD TESTS THE STACK VALUE AND IF IT'S ZERO THE LOOP IS REPEATED. FOR EXAMPLE:

```
: EX 5 BEGIN 1 - DUP NOT END DROP ;
```

FIRST THE VALUE 5 IS PUSHED ONTO THE TOP OF THE STACK. THE WORD "BEGIN" INDICATES THE BEGINNING OF A LOOP. EVERYTHING BETWEEN THE WORDS "BEGIN" AND "END" WILL BE REPEATED UNTIL THE WORD "END" FINDS A NON-ZERO VALUE ON THE STACK. THE VALUE ON TOP OF THE STACK IS ALWAYS REMOVED BY THE WORD "END". THE FIRST THING THE LOOP DOES IS PUSH A 1 ONTO THE STACK. THEN THE 1 IS SUBTRACTED FROM THE 5 BY THE WORD "-". THE ONLY THING ON THE STACK NOW IS THE VALUE 4. THE WORD "DUP" DUPLICATES THE TOP VALUE ON THE STACK. THIS IS NECESSARY BECAUSE THE WORD "END" IS GOING TO REMOVE ONE VALUE. SO NOW, AFTER THE "DUP", THE STACK CONTAINS TWO VALUES OF 4. THE "NOT" WORD SWITCHES THE TOP VALUE ON THE STACK TO 0 (HAD IT ALREADY BEEN 0, NOT WOULD HAVE SWITCHED IT TO 1). THE "END" WORD TESTS THE TOP VALUE ON THE STACK FOR 0 AND THEN REMOVES THAT VALUE. THE STACK NOW CONTAINS ONLY THE VALUE 4. SINCE THE "END" WORD FOUND A NON-ZERO VALUE, THE LOOP IS REPEATED. THIS PROCESS WILL CONTINUE TO DECREMENT THE VALUE ON THE STACK UNTIL IT REACHES ZERO. WHEN IT BECOMES 0, THE NOT WORD WILL SWITCH IT TO 1. WHEN "END" FINDS THE 1, ITS SEARCH WILL BE SATISFIED; IT WILL REMOVE THE 1, AND THE WORD AFTER THE "END" WILL BE EXECUTED. THE WORD "DROP" REMOVES THE TOP VALUE ON THE STACK, IT IS USED IN THIS EXAMPLE TO REMOVE THE RESIDUAL ZERO (0) LEFT ON THE STACK. IT'S IMPORTANT TO LEAVE THE STACK AS YOU FOUND IT. REMEMBER, NONE OF THIS WILL HAPPEN UNTIL THE WORD "EX" IS SUBSEQUENTLY FOUND IN THE INPUT STREAM AND EXECUTED.

AN ENDLESS LOOP CAN BE CREATED BY THE FOLLOWING:

```
: X1 BEGIN 0 END ;
```

OTHER WORDS CAN BE PLACED BETWEEN THE "BEGIN" AND THE "0" TO GIVE THE ENDLESS LOOP MORE PURPOSE. ONCE THE WORD X1 IS EXECUTED, IT CAN ONLY BE STOPPED BY RESETTING THE COMPUTER. THE PROGRAM MUST THEN BE "SOFTSTARTED".

6502FORTH CONTAINS A LOOPING FACILITY THAT IS VERY MUCH LIKE THE FORTRAN DO-LOOP CONSTRUCT. THE WORDS DO, LOOP, AND +LOOP ARE USED TO DEFINE THE FORTH DO-LOOP FACILITY. THE FOLLOWING EXAMPLE WILL ILLUSTRATE THE USE OF THESE NEW WORDS:

```
: EX2 4 0 DO I . LOOP ;
```

WHEN THE WORD EX2 IS EXECUTED, THE CONSTANTS 4 AND 0 WILL BE PUSHED RESPECTIVELY ONTO THE STACK. THE WORD DO USES THESE TOP TWO VALUES AS THE LIMIT AND THE INITIAL INDEX OF THE LOOP. THESE NUMBERS WILL BE REMOVED FORM THE NORMAL STACK AND PUSHED ONTO THE RETURN STACK. THEN THE WORDS AFTER THE WORD DO ARE EXECUTED. THE WORD I COPIES THE INDEX VALUE FROM THE RETURN STACK AND PUSHES IT BACK ONTO THE NORMAL STACK. (THE INDEX VALUE IS NOW ON TOP OF BOTH STACKS). THE WORD "." (PERIOD) REMOVES THE TOP VALUE ON THE NORMAL STACK AND PRINTS IT. THE WORD LOOP INCREMENTS THE INDEX VALUE (ON TOP OF THE RETURN STACK) AND THEN TESTS IT WITH THE LIMIT VALUE (SECOND VALUE ON THE RETURN STACK). IF THE NEW INDEX VALUE IS LESS THAN THE LIMIT, THEN CONTROL RETURNS TO THE FIRST WORD AFTER THE DO. OTHERWISE, THE INDEX AND LIMIT ARE POPPED FROM THE RETURN STACK AND CONTROL PASSES OUT OF THE LOOP. THE OUTPUT PRODUCED BY EXECUTION OF THE WORD EX2 IS:

```
0 1 2 3
```

NOTE THAT THE LIMIT GIVES THE NUMBER OF TIMES THE LOOP IS EXECUTED WHEN THE INITIAL INDEX IS SET TO ZERO (0). THE RANGE OF A DO LOOP IS ALWAYS EXECUTED AT LEAST ONCE. SINCE DO LOOPS USE THE RETURN STACK, CARE MUST BE TAKEN IF THE WORDS WITHIN THE LOOP ALSO USE THE RETURN STACK.

LOOPS THAT INCREMENT THE INDEX BY A VALUE OTHER THAN +1 ARE ACCOMPLISHED WITH THE +LOOP WORD. +LOOP IS LIKE LOOP, EXCEPT THAT THE CURRENT NORMAL STACK VALUE IS USED TO DETERMINE THE NEW INDEX. IF THE CURRENT NORMAL STACK VALUE IS NEGATIVE, THEN LOOPING CONTINUES UNTIL THE NEW INDEX BECOMES LESS THAN THE LIMIT VALUE.

FORWARD CONDITIONAL BRANCHES MAY BE MADE IN 6502FORTH USING THE WORDS IF, THEN, AND ELSE. THESE WORDS ARE MORE EASILY EXPLAINED THROUGH THE USE OF AN EXAMPLE:

```
: EX3 IF (TRUE-WORDS) ELSE (FALSE WORDS) THEN ;
```

WHEN EX3 IS EXECUTED, THE WORD IF TESTS THE CURRENT STACK VALUE. IF THE VALUE IS NON-ZERO, THEN THE "TRUE-WORDS" ARE EXECUTED. IF THE VALUE IS ZERO (0) THEN THE "FALSE-WORDS" ARE EXECUTED. THE WORD THEN INDICATES THE END OF THE IF...ELSE CONSTRUCT. IT IS ALWAYS REQUIRED. THE WORD ELSE IS OPTIONAL. FOR EXAMPLE:

```
: EX4 IF (TRUE-WORDS) THEN ;
```

EX4 WILL CAUSE THE "TRUE-WORDS" TO BE EXECUTED ONLY IF THE TOP STACK VALUE IS NON-ZERO.

10. ERROR CHECKING.

6502FORTH PRODUCES VARIOUS ERROR INDICATIONS. ANYTIME THAT AN ERROR IS FOUND, THE SYSTEM PRINTS THE OFFENDING WORD FOLLOWED BY A "?". A TEXT ERROR MESSAGE IS ALSO PRINTED INDICATING THE POSSIBLE CAUSE FOR THE ERROR. WHEN AN ERROR MESSAGE IS PRODUCED, 6502FORTH STOPS ITS CURRENT ACTIVITY AND RETURNS TO THE INPUT MODE. IF A WORD WAS IN THE PROCESS OF BEING DEFINED, THE DICTIONARY IS RETURNED TO ITS STATE PRIOR TO THE BEGINNING OF THE DEFINITION. THE STACKS ARE RE-INITIALIZED. THE EFFECT IS VERY SIMILAR SIMILAR TO A 6502FORTH "SOFTSTART".

11. STANDARD VOCABULARY.

FOLLOWING EACH WORD DEFINITION IS A GRAPHIC DEMONSTRATION OF THE WORD'S EFFECT ON THE NORMAL STACK. (1 2 3 4 WORD 1 2 3) SHOWS THAT THE VALUES 1, 2, 3, AND 4 WERE ENTERED, WITH 4 ON TOP OF THE STACK. THEN "WORD" WAS ENTERED. IN THIS CASE "WORD" CAUSED THE TOP STACK VALUE TO BE "POPPED" LEAVING THE 3 AS THE TOP VALUE.

TYPING WORDS & UTILITY WORDS

. (PERIOD)
CONVERT AND TYPE THE SIGNED VALUE ON TOP OF THE STACK ACCORDING TO THE CURRENT RADIX. (1 2 3 4 . 1 2 3)

#DIG
A VARIABLE WHOSE VALUE SETS THE TABULATION STOPS FOR THE WORD "." (PERIOD). THE NUMBER OF SPACES AND DIGITS TYPED IS EQUAL TO THE VALUE OF THE VARIABLE #DIG. IF #DIG IS TOO SMALL TO ALLOW COMPLETE TYPING OF A NUMBER, IT IS THEN IGNORED. (STACK IS UN-AFFECTED)

?
USES THE TOP OF THE STACK AS AN ADDRESS; AND TYPES THE VALUE AT THAT ADDRESS ACCORDING TO THE CURRENT RADIX.
(1 2 3 4 ? 1 2 3)

ECHO
TYPES THE LSB ON TOP OF THE STACK IN ASCII. IF THE CHARACTER IS NON-PRINTING, A SPACE WILL USUALLY BE PRINTED.
(1 2 3 4 ECHO 1 2 3)

SPACE
TYPES A SINGLE SPACE. (STACK IS UN-AFFECTED)

SPCES
TYPES THE NUMBER OF SPACES SPECIFIED BY THE NUMBER ON TOP OF THE STACK.
(1 2 3 4 SPCES 1 2 3)

$MSG
TYPES A STRING OF CHARACTERS UNTIL A $04 HEX VALUE IS FOUND. THE TOP STACK VALUE PROVIDES THE ADDRESS OF THE FIRST BYTE OF THE STRING TO BE TYPED. CONTROL CHARACTERS MAY BE IMBEDDED IN THE STRING.
(1 2 3 4 $MSG 1 2 3)

TYPE TYPES A STRING OF CHARACTERS WHOSE ADDRESS IS FOUND AS THE SECOND VALUE ON THE STACK. THE TOP STACK VALUE CONTAINS THE NUMBER OF CHARACTERS TO BE TYPED. (1 2 3 4 TYPE 1 2)

CR
OUTPUT A CARRIAGE RETURN. (STACK IS UN-AFFECTED)

PAGE
CLEARS A CRT DISPLAY DEVICE TC BLANKS. PAGE SHOULD NOT BE USED FOR HARD-COPY DEVICES. (STACK IS UN-AFFECTED)

?BASE
TYPES THE CURRENT RADIX USING DECIMAL AS A TEMPORARY RADIX FOR CONVERSION OF THE TYPED NUMBER. (STACK IS UN-AFFECTED)

HELP
TYPES EVERY WORD IN THE 6502FORTH DICTIONARY IN THE ORDER IT IS SEARCHED. THE ADDRESS OF EACH WORD'S CODE IS TYPED AFTER EACH WORD. IMMEDIATE WORDS ARE DISPLAYED IN REVERSE VIDEO. A PAGE OF WORDS IS DISPLAYED AT A TIME. TO OBTAIN THE NEXT PAGE, DEPRESS THE RETURN KEY. TO EXIT THE FUNCTION, DEPRESS A CTL-C SEQUENCE.
(STACK IS UN-AFFECTED)

DLIST
WILL BEGIN LISTING ALL DICTIONARY WORDS AND CODE ADDRESSES FROM THE ADDRESS OF THE WORD ON TOP OF THE STACK. THE ADDRESS MUST BE THAT OF THE CODE AND NOT THE HEADER. (1 2 3 4 DLIST 1 2 3)

.HEX
WILL PRINT THE NUMBER ON TOP OF THE STACK AS TWO (4) UNSIGNED HEXADECIMAL DIGITS. (1 2 3 4 .HEX 1 2 3)

BELL
WILL SOUND THE SYSTEMS'S BELL OR NOICE MAKER DEVICE.
(STACK IS UN-AFFECTED)

ABORT
WILL ABORT THE PRESENT SYSTEM STATUS AND RESTORE THE SYSTEM TO A SOFTSTART CONDITION. (ALL STACKS ARE RESET)

IN#
WILL INITIALIZE INPUT PORT # SPECIFIED BY THE NUMBER ON TOP OF THE STACK. (1 2 3 4 IN# 1 2 3)

PR#
WILL INITIALIZE OUTPUT PORT # SPECIFIED BY THE NUMBER ON TOP OF THE STACK. (1 2 3 4 PR# 1 2 3)

INO
WILL SET THE INPUT TO THE KEYBOARD DEVICE. (STACK IS UN-AFFECTED)

PRO
WILL SET THE OUTPUT TO THE TERMINAL CRT DEVICE.
(STACK IS UN-AFFECTED)

$SYS
WILL UPDATE THE 6502FORTH SYSTEM POINTERS TO INCLUDE NEW ENTRIES IN THE DICTIONARY AND WILL AUTOMATICALLY SAVE THE SYSTEM TO CASSETTE. THE AUTOMATIC LOADER IS FIRST RECORDED, FOLLOWED BY THE ACTUAL SYSTEM CODE ITSELF. (STACK IS UN-AFFECTED)

$PTR
WILL SET THE PRESENT SYSTEM POINTERS TO INCLUDE NEW ENTRIES THAT HAVE BEEN ADDED TO THE DICTIONARY. (STACK IS UN-AFFECTED)

$HEX WILL PRINT THE CONTENTS OF THE A REGISTER AS TWO HEXADECIMALS DIGITS. (STACK IS UN-AFFECTED)

$ERR
WILL PRINT THE ERROR MESSAGE TEXT AND CODE NUMBER.
(STACK IS UN-AFFECTED)

$SOF
WILL PRINT THE SOFTSTART MESSAGE TEXT ONLY. (STACK IS RESET)

BRK
WILL CAUSE A MACHINE BREAK TO OCCUR BACK TO THE SYSTEM MONITOR. A DISPLAY OF THE REGISTERS IS PRINTED.

BLOCK I/O & EDITOR WORDS

CASS
INITIALIZES THE CASSETTE I/O ROUTINES WITH THE BEGINNING AND ENDING VALUES, OF THE BLOCK TO BE OUTPUT, USING THE SECOND AND TOP VALUES FROM THE STACK. (1 2 3 4 5 6 LOBF 1 2 3 4)

CASW
WRITES TO CASSETTE. (STACK IS UN-AFFECTED)

CASR
READS FROM CASSETTE. (STACK IS UN-AFFECTED)

SAVE
WRITES THE BUFFER SPECIFIED BY THE NUMBER ON TOP OF THE STACK TO CASSETTE. (1 2 3 4 SAVE 1 2 3)

LOAD
READS FROM CASSETTE AND INTO THE BUFFER SPECIFIED BY THE NUMBER ON TOP OF THE STACK. (1 2 3 4 LOAD 1 2 3)

EXEC
CAUSES THE CONTENTS OF THE BUFFER SPECIFIED BY THE NUMBER ON TOP OF THE STACK TO BE PASSED THROUGH THE KEY-IN ROUTINES JUST AS IF IT WAS BEING RECEIVED FROM THE KEYBOARD. THE SCAN BEGINS WITH THE START OF THE BUFFER AND CONTINUES UNTIL THE ";S" WORD IS FOUND.
(1 2 3 4 EXEC 1 2 3)

;S
USED IN CONJUNCTION WITH EXEC TO INDICATE THE END OF THE SCAN. ;S RETURNS 6502FORTH TO THE NORMAL INPUT KEYBOARD MODE. A ;S IS AUTOMATICALLY PLACED AT THE END OF THE ACTIVE BUFFER ON LINE 21, HOWEVER THE ;S ITSELF IS NEVER DISPLAYED. (STACK IS UN-AFFECTED)

NOTENOTE***NOTE***NOTE***NOTE***NOTE***NOTE***NOTE***NOTE***NOTE

BEFORE USING ANY OF THE EDITOR WORDS BE SURE THAT YOU HAVE THE SYSTEM IN DECIMAL MODE OF OPERATION. IF THE ABOVE IS NOT DONE, THEN THE EDITOR WORDS WILL NOT WORK PROPERLY. THE EDITOR ITSELF ALLOWS EITHER BUFFER TO MAINTAIN A SCREEN OF 20 LINES OF 37 CHARACTERS EACH. A TEMPORARY LINE IS AUTOMATICALLY MAINTAINED BY THE SYSTEM, AND IT IS USED TO CONTAIN TEMPORARY TEXT MATERIAL THAT WILL BE USED IN INSERT AND REPLACE OPERATIONS. THE TEMPORARY LINE EXISTS AS LINE 22 OF BUFFER 0, HOWEVER, IT IS NEVER DISPLAYED ON EITHER BUFFER.

NOTENOTE***NOTE***NOTE***NOTE***NOTE***NOTE***NOTE***NOTE***NOTE

BUFFER
SETS THE ACTIVE BUFFER FOR ALL EDITOR OPERATIONS. THE VALJE ON TOP OF THE STACK IS USED AS THE BUFFER NUMBER. THE SYSTEM CURRENTLY SUPPORTS BUFFER 0 AND BUFFER 1 ONLY. (1 2 3 4 BUFFER 1 2 3)

CLRB
WILL CLEAR THE BUFFER SPECIFIED BY THE NUMBER ON TOP OF THE STACK TO SPACES. (1 2 3 4 CLRB 1 2 3)

LIST
WILL DISPLAY THE CONTENTS OF THE BUFFER SPECIFIED BY THE NUMBER ON TOP OF THE STACK ON A LINE NUMBERED BASIS.
(1 2 3 4 LIST 1 2 3)

COPY
WILL COPY THE BUFFER SPECIFIED BY THE SECOND NUMBER ON TOP OF THE STACK TO THE BUFFER SPECIFIED BY THE FIRST NUMBER ON TOP OF THE STACK.
(1 2 3 4 5 6 COPY 1 2 3 4)

DEL
WILL DELETE THE LINE NUMBER SPECIFIED BY THE NUMBER ON TOP OF THE STACK. ALL SUBSEQUENT LINES FOLLOWING THE DELETED LINE ARE MOVED UP. THE CONTENTS OF THE DELETED LINE ARE PLACED IN THE TEMPORARY LINE.
(1 2 3 4 DEL 1 2 3)

IN
WILL INSERT THE TEMPORARY LINE AFTER THE LINE NUMBER SPECIFIED BY THE NUMBER ON TOP OF THE STACK. ALL LINES FOLLOWING THE INSERTED LINE ARE SUBSEQUENTLY MOVED DOWN. A 0 LINE WILL EFFECTIVELY CAUSE AN INSERT ON LINE NUMBER 1. (1 2 3 4 IN 1 2 3)

R
WILL REPLACE THE LINE THAT IS SPECIFIED BY THE NUMBER ON TOP OF THE STACK WITH THE TEMPORARY LINE. (1 2 3 4 R 1 2 3)

LINE
WILL BEGIN AN AUTOMATIC LINE PROMPTING SEQUENCE, WHEREBY THE USER CAN ENTER TEXT INTO THE ACTIVE BUFFER. THE NUMBER ON TOP OF THE STACK SPECIFIES THE BEGINNING LINE NUMBER, WHILE THE SECOND NUMBER ON THE STACK SPECIFIES THE LAST LINE NUMBER + 1.
(1 2 3 4 5 6 LINE 1 2 3 4)

"
IS USED TO INSERT TEXT MATERIAL INTO THE TEMPORARY LINE. THE TEXT MATERIAL SHOULD BE PRECEED BY THE WORD " AND A SPACE. THE TEXT MATERIAL SHOULD BE TERMINATED WITH A CORRESPONDING ". THE MAXIMUM NUMBER OF CHARACTERS THAT THE TEMPORARY LINE WILL HOLD IS 37. (STACK IS UN-AFFECTED)

$PUT
WILL CAUSE THE ASCII CHARACTER IN THE A REGISTER TO BE DISPLAYED ON THE OUTPUT CRT DEVICE. (STACK IS UN-AFFECTED)

CPLR
WILL COPY FROM A SENDING LOCATION TO A DESTINATION LOCATION IN A LEFT TO RIGHT MANNER. THE MAXIMUM NUMBER OF BYTES THAT CAN BE COPIED IS 65535. THE USER MUST SPECIFY THE SOURCE ADDRESS, THE DESTINATION ADDRESS, AND THE LENGTH IN BYTES ON TOP OF THE STACK.
(1 2 3 4 CPLR 1)

CPRL
WILL COPY FROM A SENDING LOCATION TO A DESTINATION LOCATION IN A RIGHT TO LEFT MANNER. (I.E. SOURCE - TARGET, SOURCE-1 - TARGET-1, ETC.) THE SOURCE AND DESTINATION ADDRESSES MUST POINT TO THE RIGHTMOST BYTE OF THEIR RESPECTIVE AREAS. THE MAXIMUM NUMBER OF BYTES THAT CAN BE COPIED IS 65535. THE USER MUST SPECIFY THE SOURCE ADDRESS, THE DESTINATION ADDRESS, AND THE LENGTH IN BYTES ON TOP OF THE STACK. (1 2 3 4 CPRL 1)

CLRM
WILL FILL BEGINNING AT THE ADDRESS THAT IS SPECIFIED, FOR THE LENGTH IN BYTES SPECIFIED, WITH THE CHARACTER THAT IS SPECIFIED. THE USER MUST SPECIFY THE CHARACTER, THE ADDRESS, AND THE LENGTH IN BYTES ON TOP OF THE STACK. (1 2 3 4 CLRM 1)

WORD-DEFINING WORDS

:
BEGINS A COLON DEFINITION. THE NEXT WORD IN THE INPUT IS TAKEN AS THE NAME OF THE NEW WORD. THE INTERPRETER IS AUTOMATICALLY SET TO THE COMPILE MODE. (STACK IS UN-AFFECTED)

;
TERMINATES THE COLON DEFINITION, PLACES A "RETURN TO SUBROUTINE" (RTS) INSTRUCTION IN THE DICTIONARY, THEN SWITCHES THE SYSTEM BACK TO INTERPRET MODE. THIS WORD SHOULD ONLY BE USED WHILE IN COMPILE MODE.
(STACK IS UN-AFFECTED)

↑
USED WITHIN ":" (COLON) DEFINITIONS TO CAUSE THE WORDS FOLLOWING TO BE EXECUTED RATHER THAN COMPILED. THE EFFECT IS AS IF ALL WORDS FOLLOWING THE (UP-ARROW) WERE "IMMEDIATES".
(STACK IS UN-AFFECTED)

IMME
USED WITHIN ":" (COLON) DEFINITIONS JUST PRIOR TO THE ";" WORD. IT INDICATES, TO THE SYSTEM, THAT THE WORD WHICH IS BEING COMPILED SHOULD BE IDENTIFIED AS AN "IMMEDIATE" WORD. AN "IMMEDIATE" WORD IS ALWAYS EXECUTED WHEN IT IS ENCOUNTERED IN THE INPUT STREAM, REGARDLESS OF THE COMPILE/INTERPRET MODE OF THE SYSTEM. IT WILL ALWAYS BE TREATED AS IF IT FOLLOWED A "↑".
(STACK IS UN-AFFECTED)

CONS
DEFINES A WORD THAT WILL, WHEN EXECUTED, PUSH A CONSTANT VALUE ON THE STACK. THE VALUE OF THE CONSTANT IS OBTAINED FROM THE TOP OF THE STACK WHEN CONS IS EXECUTED. CONS IS NOT NORMALLY USED IN A ":" (COLON) DEFINITION. (1 2 3 4 CONS 1 2 3)

VARI
DEFINES A WORD THAT WILL, WHEN EXECUTED, PUSH AN ADDRESS ON THE STACK. THE ADDRESS POINTS TO A VALUE IN THE DICTIONARY. THE ADDRESS CAN THEN BE USED TO REFERENCE OR CHANGE THE VALUE. WHEN VARI IS EXECUTED, THE TOP STACK VALUE BECOMES THE INITIAL VALUE OF THE VALIABLE. VARI IS NOT NORMALLY USED IN A ":" (COLON) DEFINITION.
(1 2 3 4 VARI 1 2 3)

ARRAY
SETS ASIDE A REGION IN THE DICTIONARY WHOSE LENGTH (IN 16-BIT WORDS) IS THE TOP STACK VALUE. THE NAME OF THE ARRAY FOLLOWS THE WORD ARRAY IN THE INPUT STREAM. WHEN THE NEW WORD IS SUBSEQUENTLY EXECUTED, THE NUMBER ON TOP OF THE STACK IS USED AS THE INDEX TO THE ARRAY AND THE SYSTEM WILL RETURN THE ADDRESS OF THAT ELEMENT.
(1 2 3 4 ARRAY 1 2 3)

CONSTANTS.

BASE
ADDRESS OF THE CURRENT RADIX (OR BASE) VALUE.
(1 2 3 4 BASE 1 2 3 4 5)

H
ADDRESS OF THE POINTER TO THE NEXT AVAILABLE DICTIONARY LOCATION.
(1 2 3 4 H 1 2 3 4 5)

SP
ADDRESS OF THE POINTER TO THE CURRENT (OR TOP) NORMAL STACK VALUE.
(1 2 3 4 SP 1 2 3 4 5)

RS
ADDRESS OF THE POINTER TO THE CURRENT (OR TOP) RETURN STACK VALUE.
(1 2 3 4 RS 1 2 3 4 5)

LINK
ADDRESS OF THE FIRST BYTE OF THE HEADER PORTION OF THE LAST HEADER IN THE DICTIONARY. (1 2 3 4 LINK 1 2 3 4 5)

DELIM
ADDRESS OF THE DELIMITER CHARACTER BEING USED IN PARSING THE SOURCE STATEMENTS. (1 2 3 4 DELIM 1 2 3 4 5)

ERR
ADDRESS OF THE BYTE THAT CONTAINS THE ERROR CODE FOR THE ERROR PRINTING ROUTINE. (1 2 3 4 ERR 1 2 3 4 5)

EBLK
ADDRESS OF THE CURRENT ACTIVE EDIT BUFFER.
(1 2 3 4 EBLK 1 2 3 4 5)

IBP
ADDRESS OF THE POINTER TO THE INPUT BUFFER.
(1 2 3 4 IBP 1 2 3 4 5)

NORMAL STACK OPERATIONS.

DROP
REMOVES (OR POPS) THE CURRENT NORMAL STACK VALUE. THE SECOND STACK VALUE BECOMES THE CURRENT VALUE. (1 2 3 4 DROP 1 2 3)

DUP
DUPLICATES THE CURRENT, NORMAL STACK VALUE. AFTER DUP, THE CURRENT AND SECOND STACK VALUES ARE THE SAME.
(1 2 3 4 DUP 1 2 3 4 4)

-DUP
DUPLICATES THE TOP VALUE ON THE STACK, ONLY IF THE VALUE IS NOT EQUAL TO ZERO. OTHERWISE, THE TOP VALUE IS REMOVED.
(1 2 3 4 -DUP 1 2 3 4)

OVER
DUPLICATES THE SECOND STACK VALUE AND PUSHES IT ON THE STACK. THE CURRENT NORMAL STACK VALUE BECOMES THE SECOND STACK VALUE AND THE TOP AND THIRD VALUES ARE THE SAME.
(1 2 3 4 OVER 1 2 3 4 3)

ROT
MOVES THE THIRD STACK VALUE TO THE TOP OF THE STACK. THE TOP STACK VALUE BECOMES THE SECOND AND THE SECOND STACK VALUE BECOMES THE THIRD.
(1 2 3 4 ROT 1 3 4 2)

SWAP
INTERCHANGES THE TOP NORMAL STACK VALUE WITH THE SECOND VALUE.
(1 2 3 4 SWAP 1 2 4 3)

RETURN STACK OPERATIONS.

R|
RESETS THE RETURN STACK TO ITS NORMAL CONDITION.
(NORMAL STACK IS UN-AFFECTED)

<R
REMOVES THE CURRENT NORMAL STACK VALUE AND PUSHES IT ON THE RETURN STACK. (1 2 3 4 <R 1 2 3)

R>
REMOVES THE CURRENT RETURN STACK VALUE AND PUSHES IT ON THE NORMAL STACK. (1 2 3 4 R> 1 2 3 4 5)

I
PUSHES THE CURRENT RETURN STACK VALUE ON THE NORMAL STACK WITHOUT REMOVING IT FROM THE RETURN STACK. AFTER "I" THE CURRENT VALUES ON BOTH STACKS ARE THE SAME. (1 2 3 4 I 1 2 3 4 5)

J
PUSHES THE INDEX OF THE NEXT OUTER MOST DO ON TOP OF THE NORMAL STACK.
(1 2 3 4 J 1 2 3 4 5)

DICTIONARY OPERATIONS.

HERE
PUSHES THE ADDRESS OF THE NEXT AVAILABLE DICTIONARY LOCATION ON THE NORMAL STACK. (1 2 3 4 HERE 1 2 3 4 5)

TRUNC
TRUNCATES (OR DELETES) THE DICTIONARY BEGINNING WITH THE ADDRESS THAT IS FOUND ON THE TOP OF THE NORMAL STACK. THE ADDRESS ON TOP OF THE STACK MUST BE THE ADDRESS OF THE CODE PORTION OF A WORD.
(1 2 3 4 TRUNC 1 2 3)

FORGET
WILL TRUNCATE (OR DELETE) THE DICTIONARY BEGINNING WITH THE WORD THAT IS DIRECTLY FOLLOWING THE FORGET WORD ITSELF.
(STACK IS UN-AFFECTED)

,
PLACES THE CURRENT NORMAL STACK VALUE (16-BITS) INTO THE NEXT TWO AVAILABLE DICTIONARY LOCATIONS. (1 2 3 4 , 1 2 3)

C,
PLACES THE LSB (8-BITS) OF THE CURRENT NORMAL STACK VALUE INTO THE NEXT AVAILABLE DICTIONARY LOCATION. (1 2 3 4 C, 1 2 3)

D=
PLACES THE MACHINE'S ACCUMULATOR (REGISTER A) INTO THE NEXT AVAILABLE DICTIONARY LOCATION (8 BITS). (STACK IS UN-AFFECTED)

CONDITIONAL BRANCHES.

NONE OF THE FOLLOWING GROUP OF WORDS CAN BE USED OUTSIDE OF A ":" (COLON) DEFINITION. THE STACK DESCRIPTIONS SHOW THE EFFECT ON THE STACK WHEN THE WORD, THAT IS BEING DEFINED, IS SUBSEQUENTLY EXECUTED. IT IS NOT THE EFFECT DURING COMPILATION.

BEGIN
STARTS A LOOP WHICH WILL BE TERMINATED BY END. BEGIN CAN ONLY BE USED WITHIN A ":" (COLON) DEFINITION. (STACK IS UN-AFFECTED)

END
TERMINATES A BEGIN/END LOOP. END CAN ONLY BE USED WITHIN A ":" (COLON) DEFINITION. WHEN THE WORD THAT IS BEING DEFINED IS SUBSEQUENTLY EXECUTED, THE CODE GENERATED BY END WILL POP THE CURRENT, NORMAL STACK VALUE AND TEST IT FOR ZERO. IF IT IS ZERO, A BRANCH WILL BE TAKEN BACK TO BEGIN.
(1 2 3 4 END 1 2 3)

DO
STARTS A LOOP WHICH WILL BE TERMINATED BY EITHER LOOP OR +LOOP. DO CAN ONLY BE USED WITHIN A ":" (COLON) DEFINATION. WHEN THE WORD THAT IS BEING DEFINED IS SUBSEQUENTLY EXECUTED, THE DO USES THE CURRENT STACK VALUE FOR THE LOOP INDEX AND THE SECOND STACK VALUE AS THE FINAL LOOP INDEX. THESE VALUES ARE PUSHED ONTO THE RETURN STACK SO THAT THE CURRENT, RETURN STACK VALUE IS THE LOOP INDEX.
(1 2 3 4 DO 1 2)

LOOP
TERMINATES A DO/LOOP CONSTRUCT. EXECUTION OF THE LOOP WORD WILL ADD ONE TO THE LOOP INDEX. IF THE NEW INDEX VALUE IS LESS THAN THE FINAL INDEX VALUE (SECOND VALUE ON THE RETURN STACK), THEN CONTROL IS TRANSFERRED TO THE WORD FOLLOWING THE DO, AND THE LOOP IS REPEATED. OTHERWISE, THE TWO INDEX VALUES ARE POPPED FROM THE RETURN STACK AND THE LOOPING SEQUENCE ENDS. THE DO/LOOP CONSTRUCT IS ONLY USED WITHIN ":" (COLON) DEFINITIONS. (NORMAL STACK IS UN-AFFECTED)

+LOOP
TERMINATES A DO/+LOOP CONSTRUCT, AND IS EXACTLY LIKE THE DO/LOOP PREVIOUSLY DEFINED. THE EXCEPTION IS THAT THE CURRENT, NORMAL STACK VALUE IS ADDED TO THE LOOP INDEX (ON THE RETURN STACK) TO FORM THE NEW LOOP INDEX. IF THE CURRENT, NORMAL STACK VALUE IS NEGATIVE, LOOPING WILL CONTINUE WHILE THE NEW INDEX VALUE IS GREATER THAN THE FINAL VALUE (SECOND VALUE ON THE RETURN STACK). THE DO/+LOOP CONSTRUCT IS ONLY USED WITHIN ":" (COLON) DEFINITIONS.
(1 2 3 4 +LOOP 1 2 3)

IF
STARTS AN IF/THEN CONDITIONAL BRANCH. THE IF WORD, WHEN EXECUTED, TESTS (AND REMOVES) THE CURRENT, NORMAL STACK VALUE. IF THE VALUE IS NOT EQUAL TO ZERO, THEN THE CLAUSE IMMEDIATELY FOLLOWING THE IF IS EXECUTED. IF THE VALUE IS ZERO, THEN THE CLAUSE FOLLOWING THE ELSE IS EXECUTED. IF THE ELSE IS OMITTED, THEN CONTROL IS GIVEN TO THE CLAUSE FOLLOWING THE THEN. THEN IS ALWAYS REQUIRED WITH THE IF WORD. IF CAN ONLY BE USED WITHIN ":" (COLON) DEFINITIONS.
(1 2 3 4 IF 1 2 3)

ELSE
INDICATES THE BEGINNING OF A FALSE (EQUAL TO ZERO) CLAUSE IN AN IF/ELSE/THEN CONSTRUCT. ELSE IS ONLY USED WITHIN ":" (COLON) DEFINITIONS, AND ONLY WITH IF. ITS USE WITH IF IS OPTIONAL.
(STACK IS UN-AFFECTED)

THEN
TERMINATES THE IF/THEN CONSTRUCT. WORDS FOLLOWING THEN ARE EXECUTED WITHOUT RESPECT TO THE PRECEEDING IF CONDITIONAL TEST. THEN IS ONLY USED WITHIN ":" (COLON) DEFINITIONS, AND ONLY WITH IF. IF CANNOT BE USED WITHOUT A TERMINATING THEN. IF A WORD IS DEFINED USING IF WITHOUT THEN, RESULTS WILL BE UN-PREDICTABLE.
(STACK IS UN-AFFECTED)

LEAVE
SETS THE INDEX IN A LOOP TO THE FINAL VALUE OF THE LOOP, SO THAT THE LOOP CAN BE TERMINATED SOONER. (STACK IS UN-AFFECTED)

ADDRESS OPERATORS.

@
USES THE CURRENT STACK VALUE AS AN ADDRESS, AND PLACES THE 16-BIT VALUE AT THAT ADDRESS ON THE STACK. THE ADDRESS THAT WAS ON THE STACK IS REPLACED BY ITS VALUE. (1 2 3 4 @ 1 2 3 4)

C@
USES THE CURRENT STACK VALUE AS AN ADDRESS AND PLACES THE 8-BIT LSB OF THE 16-BIT VALUE AT THAT ADDRESS ON THE STACK. THE ADDRESS THAT WAS ON THE STACK IS REPLACED BY THE 8-BIT VALUE.
(1 2 3 4 C@ 1 2 3 4)

!
USES THE CURRENT STACK VALUE AS AN ADDRESS AND STORES THE SECOND (16-BIT) STACK VALUE AT THAT ADDRESS. (1 2 3 4 ! 1 2)

C!
USES THE CURRENT STACK VALUE AS AN ADDRESS AND STORES THE (8-BIT) LSB OF THE SECOND STACK VALUE AT THAT ADDRESS. (1 2 3 4 C! 1 2)

+!
USES THE CURRENT STACK VALUE AS AN ADDRESS. THE VALUE AT THAT ADDRESS IS REPLACED BY THE SUM OF ITSELF AND THE SECOND STACK VALUE. BOTH THE ADDRESS AND THE ADDEND ARE REMOVED FROM THE STACK.
(1 2 3 4 +! 1 2)

-!
USES THE CURRENT STACK VALUE AS AN ADDRESS. THE VALUE AT THAT ADDRESS IS REPLACED BY THE DIFFERENCE BETWEEN ITSELF AND THE SECOND VALUE ON TOP OF THE STACK. BOTH ENTRIES ON TOP OF THE STACK ARE REMOVED.
(1 2 3 4 -! 1 2)

LOGICAL OPERATORS.

AND
PERFORMS A LOGICAL "AND" WITH THE TOP TWO STACK VALUES. THE TWO VALUES ARE REPLACED, ON THE STACK, BY THE RESULT.
(1 2 3 4 AND 1 2 3)

OR
PERFORMS A LOGICAL "OR" WITH THE TOP TWO STACK VALUES. THE TWO VALUES ARE REPLACED, ON THE STACK, BY THE RESULT.
(1 2 3 4 OR 1 2 3)

XOR
PERFORMS A LOGICAL "EXCLUSIVE OR" WITH THE TOP TWO STACK VALUES. THE TWO VALUES ARE REPLACED, ON THE STACK, BY THE RESULT.
(1 2 3 4 XOR 1 2 3)

SWAB
EXCHANGES THE LSB WITH THE MSB OF THE CURRENT, STACK VALUE.
(1 2 3 4 SWAB 1 2 3 4)

COM
PERFORMS A 1'S COMPLEMENT WITH THE VALUE ON THE TOP OF THE STACK. THE COMPLEMENTED VALUE IS SUBSTITUTED WITH THE ORIGINAL VALUE.
(1 2 3 4 COM 1 2 3 4)

NOR
PERFORMS A LOGICAL "NOR" WITH THE TOP TWO STACK VALUES. THE TWO VALUES ARE REPLACED, ON THE STACK, BY THE RESULT.
(1 2 3 4 NOR 1 2 3 4)

NAND
PERFORMS A LOGICAL "NAND" WITH THE TOP TWO STACK VALUES. THE TWO VALUES ARE REPLACED, ON THE STACK, BY THE RESULT.
(1 2 3 4 NAND 1 2 3 4)

TEST OPERATORS.

0=
TESTS THE CURRENT STACK VALUE FOR ZERO. IF THE VALUE IS ZERO, IT IS REPLACED ON THE STACK WITH THE VALUE ONE (1). A NON-ZERO VALUE IS REPLACED WITH THE VALUE ZERO. (1 2 3 4 0= 1 2 3 4)

NOT
IDENTICAL TO 0= (ABOVE). (1 2 3 4 NOT 1 2 3 4)

0<
TESTS THE CURRENT STACK VALUE FOR NEGATIVE. IF THE VALUE IS NEGATIVE, IT IS REPLACED ON THE STACK WITH THE VALUE ONE (1). A POSITIVE (OR ZERO) VALUE IS REPLACED BY THE VALUE ZERO.
(1 2 3 4 0< 1 2 3 4)

=
TESTS THE TOP TWO STACK VALUES FOR EQUALITY. IF THEY ARE EQUAL, BOTH VALUES ARE REPLACED BY THE VALUE ONE (1). OTHERWISE, BOTH VALUES ARE REPLACED BY THE VALUE ZERO. (1 2 3 4 = 1 2 3)

>
TESTS THE SECOND STACK VALUE FOR GREATER THAN THE TOP STACK VALUE. A SIGNED COMPARISON IS USED. IF THE SECOND STACK VALUE IS GREATER, BOTH VALUES ARE REPLACED BY THE VALUE ONE (1). OTHERWISE, BOTH VALUES ARE REPLACED BY THE VALUE ZERO.
(1 2 3 4 > 1 2 3)

<
TESTS THE SECOND STACK VALUE FOR LESS THAN THE TOP STACK VALUE. A SIGNED COMPARISON IS USED. IF THE SECOND STACK VALUE IS LESS, BOTH VALUES ARE REPLACED BY THE VALUE ONE (1). OTHERWISE, BOTH VALUES ARE REPLACED BY THE VALUE ZERO. (1 2 3 4 < 1 2 3)

MIN
COMPARES THE TOP TWO STACK VALUES AND KEEPS ONLY THE SMALLER VALUE. THE GREATER VALUE (OR TOP VALUE IF EQUAL) IS REMOVED FROM THE STACK. A SIGNED COMPARISON IS USED.
(1 2 3 4 MIN 1 2 3)

MAX
COMPARES THE TOP TWO STACK VALUES AND KEEPS ONLY THE LARGER VALUE. THE SMALLER VALUE (OR TOP VALUE IF EQUAL) IS REMOVED FROM THE STACK. A SIGNED COMPARISON IS USED.
(1 2 3 4 MAX 1 2 3)

ARITHMETIC OPERATORS.

+
ADDS THE TOP TWO STACK VALUES AND REPLACES THEM, ON THE STACK, WITH THEIR SUM. (1 2 3 4 + 1 2 3)

1+
INCREMENTS THE TOP STACK VALUE BY 1. (1 2 3 4 1+ 1 2 3 4)

-
SUBTRACTS THE TOP STACK VALUE FROM THE SECOND STACK VALUE AND REPLACES THEM, ON THE STACK, WITH THEIR DIFFERENCE.
(1 2 3 4 - 1 2 3)

*
MULTIPLIES THE TOP TWO STACK VALUES AND REPLACES THEM, ON THE STACK, WITH THEIR 16-BIT PRODUCT. THE MULTIPLIER MUST BE LESS THAN 128.
(1 2 3 4 * 1 2 3)

2*
DOUBLES THE TOP STACK VALUE. (1 2 3 4 2* 1 2 3 4)

/MOD
DIVIDES THE SECOND (16-BIT) STACK VALUE BY THE TOP (8-BIT) STACK VALUE. THE SECOND STACK VALUE (DIVIDEND) IS REPLACED BY THE (16-BIT) QUOTIENT. THE TOP STACK VALUE (DIVISOR) IS REPLACED BY THE (8-BIT) REMAINDER. (1 2 3 4 /MOD 1 2 3 4)

/
EXACTLY THE SAME AS A /MOD, EXCEPT THAT THE QUOTIENT IS THE ONLY VALUE THAT IS RETURNED ON TOP OF THE STACK. (1 2 3 4 / 1 2 3)

*/
PERFORMS THE ALGEBRAIC EXPRESSION ((A * B) / C). THE VALUE ON TOP OF THE STACK IS THE RESULT OF EVALUATING THE EXPRESSION.
(1 2 3 4 5 */ 1 2 3)

*/MOD
PERFORMS THE ALGEBRAIC EXPRESSION ((A * B) /MOD C). THE VALJE ON TOP OF THE STACK IS THE REMAINDER, WHILE THE SECOND VALUE ON THE STACK IS THE QUOTIENT. (1 2 3 4 5 */MOD 1 2 3 4)

MINUS
CHANGES THE SIGN OF THE TOP STACK VALUE.
(1 2 3 4 MINUS 1 2 3 4)

ABS
REPLACES THE TOP STACK VALUE WITH ITS ABSOLUTE VALUE. (NEGATIVE VALUES ARE MADE POSITIVE; POSITIVE VALUES ARE LEFT ALONE).
(1 2 3 4 ABS 1 2 3 4)

DECIMAL
CHANGES THE RADIX (OR BASE) TO DECIMAL. ARITHMETIC OPERATIONS AND ASCII CONVERSIONS ARE AFFECTED BY THE VALUE OF THE RADIX.
(STACK IS UN-AFFECTED)

HEX
CHANGES THE RADIX (OR BASE) TO HEXADECIMAL. ARITHMETIC OPERATIONS AND ASCII CONVERSIONS ARE AFFECTED BY THE VALUE OF THE RADIX.
(STACK IS UN-AFFECTED)

OCTAL
CHANGES THE RADIX (OR BASE) TO OCTAL. ARITHMETIC OPERATIONS AND ASCII CONVERSIONS ARE AFFECTED BY THE VALUE OF THE RADIX.
(STACK IS UN-AFFECTED)

MISCELLANEOUS WORDS.

'
THE DICTIONARY IS SEARCHED FOR THE WORD THAT IMMEDIATELY FOLLOWS THE ' (QUOTE). WHEN THAT WORD IS FOUND, THE ADDRESS OF THE EXECUTABLE MACHINE CODE (FOLLOWING ITS HEADER) IS PLACED ON THE STACK.
(1 2 3 4 ' WORD 1 2 3 4 5)

(
BEGINS A STRING OF TEXT THAT WILL BE SKIPPED BY THE 6502FORTH INTERPRETER. THE STRING BEING IGNORED MUST BE TERMINATED BY A ")". THE FIRST BLANK FOLLOWING THE "(" IS A 6502FORTH REQUIREMENT AND CANNOT BE OMITTED. (STACK IS UN-AFFECTED)

BYE
RETURNS CONTROL TO THE MACHINE'S MONITOR.

ST>A
WILL TAKE THE LEAST SIGNIFICANT BYTE (LSB) OFF THE STACK AND PLACE IT IN THE A REGISTER. (1 2 3 4 ST>A 1 2 3)

A>ST
WILL PLACE THE VALUE CONTAINED IN THE A-REGISTER ON TOP OF THE NORMAL STACK. (1 2 3 4 A>ST 1 2 3 4 5)

SP=
WILL PLACE THE VALUE CONTAINED IN THE A REGISTER AT THE ADDRESS SPECIFIED BY THE STACK POINTER. (STACK IS UN-AFFECTED)

STORE
WILL STORE THE VALUE CONTAINED IN THE A REGISTER AT THE NEXT AVAILABLE POSITION IN THE DICTIONARY. (STACK IS UN-AFFECTED)

NORMAL
WILL TURN OFF FLASHING OR INVERSE VIDEO MODE.
(STACK IS UN-AFFECTED)

INVERSE
WILL SET THE VIDEO OUTPUT TO BLACK ON WHITE.
(STACK IS UN-AFFECTED)

FLASH
WILL SET THE VIDEO OUTPUT TO FLASHING.
(STACK IS UN-AFFECTED)

PRINT
WILL TYPE THE STRING FOLLOWING THE "PRINT" WORD. THE STRING IS ENCLOSED WITHIN A PAIR OF UNIQUE DELIMITERS. THE FIRST NON-BLANK CHARACTER AFTER THE "PRINT" WORD DEFINES THE DELIMITER. THIS IS AN IMMEDIATE WORD AND MUST BE USED INSIDE A COLON ":" DEFINITION.
(STACK IS UN-AFFECTED)

D$
WILL OUTPUT A CTL-D. (STACK IS UN-AFFECTED)

SIZE
WILL TYPE THE SIZE OF THE 6502FORTH SYSTEM.
(STACK IS UN-AFFECTED)

XAM
TYPES 128 BYTES OF MEMORY STARTING AT THE ADDRESS SPECIFIED BY THE NUMBER ON TOP OF THE STACK.
(1 2 3 4 XAM 1 2 3)

L
DISASSEMBLES 20 INSTRUCTIONS INTO THE 6502 ASSEMBLY MNEMONIC CODE, STARTING AT THE ADDRESS SPECIFIED BY THE NUMBER ON TOP OF THE STACK.
(1 2 3 4 L 1 2 3)

C.HEX
WILL TYPE THE LOW ORDER BYTE ON TCP OF THE STACK AS TWO HEXADECIMAL DIGITS.
(1 2 3 4 C.HEX 1 2 3)

HAD
HEXADECIMAL ASCII DUMP. SIMILAR TO "XAM" WORD WITH ASCII EQUIVALENTS PRINTED UNDER THE HEX VALUES.
(1 2 3 4 HAD 1 2 3)

DISK COMMAND EXTENSION.

THE FOLLOWING GROUP OF WORDS ARE AVAILABLE TO USERS OF THE 6502FORTH SYSTEM WITH THE DISK COMMAND EXTENSION MODULES. NOTE THAT IF A DISK ERROR OCCURS, THEN THE DOS EXITS TO INTEGER BASIC, WHICH MAY DESTROY SOME OF THE 6502FORTH POINTERS IN PAGE ZERO. IF THIS OCCURS, THE ONLY SOLUTION IS TO RE-ENTER 6502FORTH AT THE HARDSTART ADDRESS.

DOS
ENABLE I/O VECTORS FOR THE DOS.
(STACK IS UN-AFFECTED)

UNDOS
DISABLES THE I/O VECTORS FOR DOS.
(STACK IS UN-AFFECTED)

OLOAD
WILL LOAD THE SCREEN THAT IS SPECIFIED BY THE NUMBER ON TOP OF THE STACK INTO BUFFER 0.
(1 2 3 4 OLOAD 1 2 3)

1LOAD
WILL LOAD THE SCREEN THAT IS SPECIFIED BY THE NUMBER ON TOP OF THE STACK INTO BUFFER 1.
(1 2 3 4 1LOAD 1 2 3)

OSAVE
WILL SAVE THE SCREEN THAT IS SPECIFIED BY THE NUMBER ON TOP OF THE STACK FROM BUFFER 0.
(1 2 3 4 OSAVE 1 2 3)

1SAVE
WILL SAVE THE SCREEN THAT IS SPECIFIED BY THE NUMBER ON TOP OF THE STACK FROM BUFFER 1.
(1 2 3 4 1SAVE 1 2 3)

FSAVE
WILL UPDATE THE 6502FORTH SYSTEM POINTERS TO INCLUDE NEW ENTRIES IN THE DICTIONARY AND WILL PROCEED TO SAVE THE ENTIRE SYSTEM TO DISK.
(STACK IS UN-AFFECTED)

LOCK
WILL LOCK THE SCREEN THAT IS SPECIFIED BY THE NUMBER ON TOP OF THE STACK.
(1 2 3 4 LOCK 1 2 3)

UNLOCK
WILL UNLOCK THE SCREEN THAT IS SPECIFIED BY THE NUMBER ON TOP OF THE STACK.
(1 2 3 4 UNLOCK 1 2 3)

DELETE
WILL DELETE THE SCREEN THAT IS SPECIFIED BY THE NUMBER ON TOP OF THE STACK.
(1 2 3 4 DELETE 1 2 3)

CATALOG
WILL PERFORM A DISK CATALOG FUNCTION.
(STACK IS UN-AFFECTED)

SLOT
A VARIABLE THAT INDICATES THE SLOT NUMBER OF THE CONTROLLER THAT THE DISK UNIT IS ATTACHED TO. NOTE SLOT IS PRE-INITIALIZED TO SIX (6).

DRIVE
A VARIABLE THAT INDICATES THE DISK DRIVE NUMBER POSITION ATTACHED TO THE CONTROLLER. NOTE THAT DRIVE IS PRE-INITIALIZED TO ONE (1).

12. 6502FORTH ASSEMBLER

A LARGE NUMBER OF THE WORDS IN THE 6502FORTH DICTIONARY CLOSELY RESEMBLE THE 6502 MNEMONIC INSTRUCTION SET. THEY ARE USED TO GENERATE 6502 MACHINE CODE AND ADD IT TO THE DICTIONARY. FOR THIS REASON, THEY NEED TO BE EXECUTED IMMEDIATELY (I.E. DURING THE COMPILATION OF SOME OTHER WORD). TO CAUSE THEM TO BE EXECUTED IMMEDIATELY, THEY USUALLY FOLLOW THE "↑" (UP-ARROW).

ALTHOUGH THE MNEMONICS ARE ALMOST THE SAME, THE SYNTAX OF THE INSTRUCTIONS IS QUITE DIFFERENT. THE TRADITIONAL "OPERATOR FOLLOWED BY OPERAND" FORMAT HAS BEEN REVERSED TO ACCOMODATE THE 6502FORTH SYNTAX. THIS ALLOWS THE OPERANDS TO BE PUSHED ONTO THE STACK AND THEN OPERATED UPON BY THE MNOMONIC OP CODES. THE USE OF THE STACK ALSO REQUIRES THAT THE 6502FORTH ASSEMBLER WORDS BE INFORMED OF THE DESIRED ADDRESSING MODE OF THOSE INSTRUCTIONS THAT CONTAIN ADDRESSES. THE 6502FORTH ASSEMBLER FORMAT IS AS FOLLOWS:

OPERAND MNEMONIC ADDRESSING MODE

MOST OF THE 6502 MNEMONIC INSTRUCTION SET HAS BEEN INCLUDED IN THE 6502FORTH DICTIONARY. ALL MNEMONICS ARE TERMINATED BY A COMMA SO THAT VALID HEX NUMBERS OR OTHER VALID 6502FORTH WORDS (E.G. BCC ADC) ARE NOT CONFUSED.

FOLLOWING IS A LIST OF ALL MNEMONIC OP CODES AVAILABLE IN THE 6502FORTH DICTIONARY. THE * (ASTERISK) FOLLOWING SOME WORDS IS NOT PART OF THE WORD. IT INDICATES WHICH OF THE WORDS REQUIRES AN ADDRESSING MODE IDENTIFIER.

THE LIST IS DIVIDED INTO THREE SECTIONS. THE WORDS IN THE FIRST SECTION REQUIRE AN OPERAND ON THE STACK AND AN ADDRESSING MODE IDENTIFIER FOLLOWING THE WORD. THIS GROUP OF WORDS WILL ADD EITHER 2 OR 3 BYTES OF MACHINE CODE (DEPENDING ON THE ADDRESSING MODE AND INSTRUCTION) TO THE DICTIONARY.

THE SECOND GROUP OF WORDS ALL IMPLY THE RELATIVE ADDRESSING MODE, SO NO MODE IDENTIFIER IS USED. THIS GROUP OF WORDS ALSO FINDS ITS OPERANDS ON THE STACK. TWO (2) BYTES OF MACHINE CODE WILL BE ADDED TO THE DICTIONARY BY ANY OF THESE WORDS.

THE THIRD GROUP OF WORDS ARE IMPLICIT, AND REQUIRE NEITHER AN OPERAND NOR AN ADDRESSING MODE IDENTIFIER. EACH OF THESE WORDS WILL ADD 1 BYTE OF MACHINE CODE TO THE DICTIONARY WHEN EXECUTED.

GROUP 1 (FORMAT: OPERAND MNEMONIC MODE)

ADC, *	AND, *	ASL, *	CMP, *
DEC, *	EOR, *	INC, *	LDA, *
LDY, *	LSR, *	ORA, *	ROL, *
SBC, *	STA, *	STY, *	

GROUP 2 (FORMAT: OPERAND MNEMONIC)

BCC,	BCS,	BEQ,	BMI,
BNE,	BPL,	JMP,	JSR,

GROUP 3 (FORMAT: MNEMONIC)

CLC,	DEX,	DEY,	INX,
INY,	RTS,	SEC,	PHA,
PLA,	SED,	CLD,	SEI,
TSX,	TXS,	CLI,	RTI,
PHP,	CLV,	NOP,	TAX,
TXA,	TAY,	TYA,	

* THE SIX ADDRESSING MODE IDENTIFIER WORD ARE:

#	IMMEDIATE
a#	ABSOLUTE ADDRESSING. ZERO PAGE ADDRESSING IS USED IF THE ADDRESS IS IN PAGE ZERO.
#A	ACCUMULATOR.
X+	ABSOLUTE X INDEXED. WILL NOT WORK FOR ADDRESS REFERENCES TO PAGE ZERO.
X)	INDEXED INDIRECT.
)Y	INDIRECT INDEXED.

IT IS THE RESPONSIBILITY OF THE USER TO INSURE THAT ONLY VALID ADDRESSING MODES ARE USED.

SYSTEM ERROR CODES

00	EXECUTION OF AN ABORT SEQUENCE
01	NORMAL STACK OVERFLOW
02	NORMAL STACK UNDERFLOW
03	RETURN STACK OVERFLOW
04	RETURN STACK UNDERFLOW
05	DICTIONARY OVERFLOW
06	INPUT BUFFER FULL
07	WRONG STATUS MODE
08	WORD NOT IN DICTIONARY ...OR... NUMBER IN WRONG BASE
09	...RESERVED...
10	MULTIPLY OVERFLOW
11	MULTIPLIER GREATER THAN 128
12	...RESERVED...
13	...RESERVED...
14	...RESERVED...
15	...RESERVED...
16	...RESERVED...
17	...RESERVED...
18	...RESERVED...
19	...RESERVED...
20	INVALID BUFFER NUMBER
21	NO LENGTH SPECIFIED FOR MOVE OPERATION
22	NEGATIVE LINE NUMBER
23	...RESERVED...
24	...RESERVED...
25	...RESERVED...

SYSTEM CONFIGURATION

- APPLE II COMPUTER SYSTEM
- 16K BYTES OF RAM MEMORY
- ONE SOURCE INPUT DEVICE (KEYBOARD)
- ONE LIST DEVICE (CRT VIDEO DISPLAY)
- ONE STORAGE DEVICE (CASSETTE)

SYSTEM LOADING INSTRUCTIONS

1. ASSURE THAT THE SYSTEM IS IN THE MONITOR STATE. THE ASTERISK "*" SHOULD BE THE SYSTEM PROMPT CHARACTER.

2. PLACE APPLEFORTH SYSTEM CASSETTE ON TAPE UNIT.

3. TYPE: 100.115R100G

4. TURN CASSETTE TAPE ON PLAY AND QUEUE ON SOUND.

5. DEPRESS THE "RETURN" KEY WHEN QUEUED ON SOUND.

6. APPLEFORTH SYSTEM WILL BE AUTOMATICALLY LOADED AND EXECUTED.

SYSTEM INITIALIZATION ADDRESSES

0800	HARDSTART ADDRESS
0803	SOFTSTART ADDRESS
0806	MONITOR INPUT ROUTINE
0809	MONITOR OUTPUT ROUTINE
080C	HOME CURSER & CLEAR CRT
080F	CARRIAGE RETURN & LINE FEED
0812	READ FROM CASSETTE
0815	WRITE TO CASSETTE
0818	RETURN TO MONITOR OR DOS

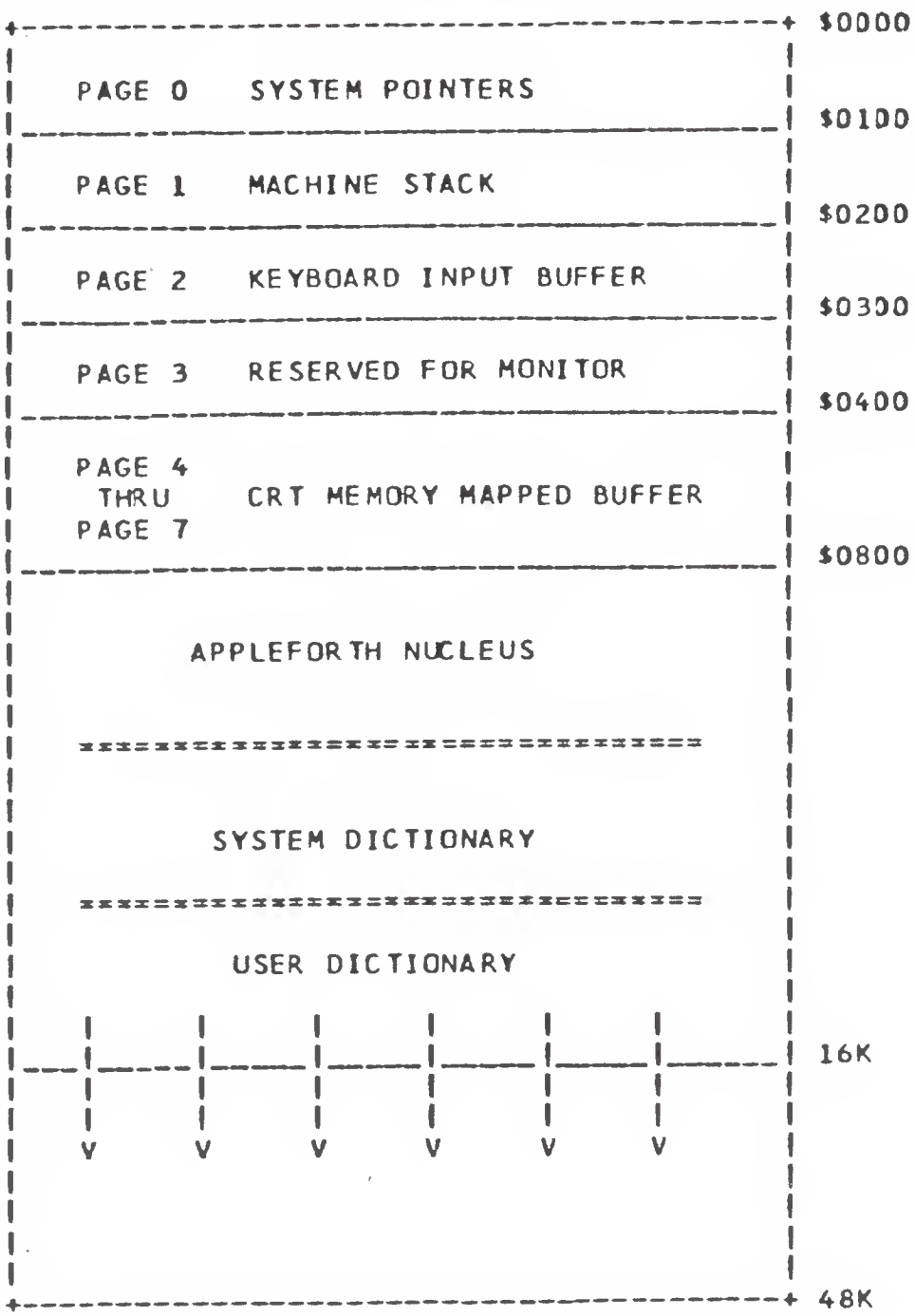
USER SELECTABLE PARAMETERS

0833	FIRST AVAILABLE BYTE OF DICTICNARY
0835	ADDRESS OF LAST HEADER IN DICTIONARY
0837	NORMAL STACK LOCATION
0839	RETURN STACK LOCATION
083B	BUFFER 0 LOCATION
083D	BUFFER 1 LOCATION
083F	INPUT BUFFER LOCATION
0841	LOGICAL DICTIONARY END

PAGE ZERO USAGE & SIGNIFICANCE

B1 - BB	WORK AREAS
D0	BASE ADDRESS OF NORMAL STACK
D2	BASE ADDRESS OF RETURN STACK
D4	BASE ADDRESS OF INPUT BUFFER
D6	NEXT AVAILABLE BYTE IN DICTIONARY
D8	ADDRESS OF LAST HEADER IN DICTIONARY
DA	ADDRESS OF ACTIVE PARSING I/O BUFFER
DC	BASE ADDRESS OF BUFFER 0
DE	BASE ADDRESS OF BUFFER 1
E0	NORMAL STACK POINTER
E1	RETURN STACK POINTER
E3	BASE OR RADIX
E4	SYSTEM STATUS
E5	DELIMITER
E6	SIGN
E7	IMMEDIATE WORD INDICATOR
E8	Y REGISTER SAVE
E9	X REGISTER SAVE
EA	...RESERVED...
EB	...RESERVED...
EC	...RESERVED...
ED - F0	WORD BUILD AREA
F1	ERROR BYTE INDICATOR
F2 - FF	...RESERVED...

S Y S T E M M E M O R Y M A P



14. EXAMPLES.

The following series of examples assume that the 6502FORTH System is operating in Decimal mode or Base 10.

EXAMPLE 1.

```
DECIMAL

( EX 1: ASCII CHARACTER SET IN ASCENDING SEQUENCE )

: EX1 64 0 DO 32 I + ECHO SPACE LOOP ;

CR EX1
```

EXAMPLE 2.

```
( EX 2: ASCII CHARACTER SET IN DESCENDING SEQUENCE )

: EX2 64 0 DO 95 I - ECHO SPACE LOOP ;

CR EX2
```

EXAMPLE 3.

```
( EX 3: COUNT FROM 1 TO 100 IN BASE 10 )

: EX3 101 1 DO I . LOOP ;

CR EX3
```

EXAMPLE 4.

```
( EX 4: COMPOSE A LINE OF FORTY DASHES )

: EX4 40 0 DO 45 ECHO LOOP ;

CR EX4
```

EXAMPLE 5.

```
( EX 5: COMPOSE TWO POSTS WITH SPACES IN BETWEEN )

: EX5 73 ECHO 38 0 DO 32 ECHO LOOP 73 ECHO ;

CR EX5
```

EXAMPLE 6.

```
( EX 6: COMPOSE A BOX USING WORDS OF EXAMPLE 4 AND 5 )

: EX6 PAGE EX4 2Ø Ø DO EX5 LOOP EX4 ;

EX6
```

EXAMPLE 7.

```
( EX 7: DISPLAY THE ENTIRE CHARACTER SET )

: CHRS PAGE 15 VTAB 1 TAB
    1Ø Ø DO
       I 1Ø * I Ø LRMAP
          OVER I + OVER I + C!
          LOOP
       DROP DROP
       LOOP ;

CHRS
```

JAMES, JOHN S.

"FORTH DUMP PROGRAMS"

DR. DOBBS JOURNAL OF COMPUTER CALISTENICS & ORTHODONTIA, NUMBER 28, PAGE 26-28, BOX E, MENLO PARK, CALIFORNIA 94025

FORTH INTEREST GROUP

"FORTH DIMENSIONS"

JUNE/JULY 1978, VOLUME 1 NUMBER 1, F.I.G., 787 OLD COUNTY ROAD, SAN CARLOS, CALIFORNIA 94070

JAMES, JOHN S.

"FORTH INTERNALS EXAMPLE"

AUGUST 7, 1978, J.S.J, 1090 MILLER AVENUE, BERKELEY, CALIFORNIA 94708

EWING, MARTIN S.

"THE CALTECH FORTH MANUAL"

SEPTEMBER 1974, OWENS VALLEY RADIO OBSERVATORY, CALIFORNIA INSTITUTE OF TECHNOLOGY, PASADENA, CALIFORNIA 91109

STEVENS, W. RICHARD

"FORTH SYSTEMS REFERENCE MANUAL"

KITT PEAK NATIONAL OBSERVATORY, TUCSON, ARIZONA 85726

FORTH, INC.

"MINI FORTH INTRODUCTION"

815 MANHATTAN AVENUE, MANHATTAN BEACH, CALIFORNIA 90266

STEIN, PHILIP

"THE FORTH DIMENSION: MINI LANGUAGE HAS MANY FACES"

COMPUTER DECISIONS, NOVEMBER 1975, PAGE 10

RATHER, ELIZABETH D.

"THE FORTH APPROACH TO OPERATING SYSTEMS"

FORTH, INC., 815 MANHATTAN AVENUE, MANHATTAN BEACH, CALIFORNIA 90266

MOORE, CHARLES H.

"FORTH: A NEW WAY TO PROGRAM A MINICOMPUTER"

ASTRONOMY ASTROPHYSICS JOURNAL SUPPLEMENT, NUMBER 15, PAGES 497-511

RATHER, E.D. & MOORE, C.H.

"FORTH HIGHLEVEL PROGRAMMING TECHNIQUE ON MICROPROCESSORS"

PAPER PRESENTED AT ELECTRO 76 PROFESSIONAL PROGRAM, BOSTON, MASSACHUSETTS, MAY 11-14, 1976

MOORE, CHARLES H.

"THE FORTH PROGRAM FOR SPECTRAL LINE OBSERVING"

PROCEEDINGS OF I.E.E.E., VOLUME 61, SEPTEMBER 1973, PAGES 1346-1349

RATHER, ELIZABETH D.

"FORTH: A FRESH APPROACH TO PROGRAMMING"

FORTH, INC.; 815 MANHATTAN AVENUE, MANHATTAN BEACH, CALIFORNIA 90266

FORTH, INC.

"MICROFORTH PRIMER"

FORTH, INC.; 815 MANHATTAN AVENUE, MANHATTAN BEACH, CALIFORNIA 90266

HARRIS, KIM

"EXTENSIBILITY WITH FORTH"

1978

THE DIGITAL GROUP, INC.

"CONVERS"

THE DIGITAL GROUP, INC., P.O. BOX 6528, DENVER, CO 80206

JAMES, JOHN

"FORTH FOR MICROCOMPUTERS"

1090 MILLER AVENUE, BERKELEY, CALIFORNIA 94708

DECUS

"FORTH PROGRAMMING SYSTEM FOR THE PDP-11"

DECUS NO. 11-232, SEPT, 1975

SINCLAIR, W.S.

"FORTH: A STACK ORIENTED LANGUAGE"

INTERFACE AGE, SEPT. 1976

RATHER, E.D. & MOORE, C.H.

"THE USE OF FORTH IN PROCESS CONTROL"

PRESENTED AT INTERNATIONAL '77 MINI-MICRO COMPUTER CONFERENCE

GENEVA, ITALY, MAY 26TH, 1977

PROGRAMMA CONSULTANTS

"6502FORTH: PRELIMINARY USERS MANUAL"

PROGRAMMA CONSULTANTS, 3400 WILSHIRE BOULEVARD, LOS ANGELES, CA 90010